Block 4: Learning BN Structure

1. Introduction.
2. Score-Based Structure Learning.
   a) The Bayesian Information Criterion (BIC) score.
   b) An heuristic DAG search algorithm: the Greedy Search-And-Score Algorithm.
   c) Score-Based Structure Learning with bnlearn.
3. Constraint-Based Structure Learning.
4. Hybrid Structure Learning.

1. Introduction

We have thus far assumed that we know the structure of a BN and concerned ourselves with estimating the values of its parameters (Block 3). Our approach for this estimation has been to search for **ML estimates**, that is, ones that maximize the probability of observing the given data set.

We now assume that the structure itself is unknown and then we present some methods for learning it from a given data set. Once we learn the DAG form data, we can then learn the parameters (Block 3). The result will be a BN that we can use to do inference (Block 2).

The structure of a BN may be determined by causal reasoning, or formally constructed by engineering considerations. This is not considered here; **the problem considered here is that of locating a structure purely from data**.
This is the task of ***structure learning***. Learning Bayesian network structures has been proven to be **NP-hard** by Chickering (1996). Indeed, a simple look at the number of possible DAGs for a given number of nodes will indicate the problem is hard (see formula slide 28: for **10** nodes, approximately $\mathbf{4.2 \times 10^{18}}$ possible DAGs!).

Due to the computational complication, it is not feasible to find an optimum solution and for that reason heuristic methods are used, that is, they do not guarantee to give an optimum solution, although it is a solution that can be sufficiently satisfactory for our purposes.

Indeed, based only on data, (heuristic) structure learning methods tend to fall generally into one of the four main categories:

A. "Score-based" (also named "Search-and-score") methods, where a score function is used and the algorithm attempts to find the structure that maximises the score function.

B. "Constraint-based" methods, where conditional independence tests are carried out and independence relations thus established provide constraints that the output graph should satisfy.

C. Hybrid methods, that use constraint-based methods to reduce the search space, and a score function to find the pseudo-optimum in the small space.

D. "Dynamic programming" approach.

We focus on **complete data**. Dealing with incomplete data is similar as far as scoring functions are concerned, but is much more demanding computationally.

Additional comments

① The data set D will be randomly partitioned into:

i. a *training set* T, and
ii. a *validation set* V,

such that they are disjoint. Thus, none of the validation cases are used for training and can be considered to be new to the BN model.

After training from T, we will apply BN to the cases from V in order to determine *accuracy of predictions*.

② The *performance* of the Bayesian network training algorithms (both for parameter and for structure learning) always depends on the *sufficiency* of the training database.

And the *size* of a sufficient BN database depends on:

a. the number of nodes,
b. the size of their domain, and
c. the underlying probability distributions.

While the nodes and their domain are known from the problem formulation, the underlying probability distribution is typically unknown a priori.

2. Scored-Based Structure Learning

In score-based structure learning we assign a score to a DAG, based on how well the DAG fits the data. That is, we evaluate the goodness of fit of a candidate network structure (DAG) by means of a score function.

The candidate network structure is provided with a maximum likelihood estimate (MLE) of the conditional probability tables associated with the nodes of the network in order to compute its score. If the data is incomplete, the EM algorithm must be used. If the data is complete, this can be done by counting cases and normalizing the counts.

Different score functions can be used.

We will introduce the **Bayesian Information Criterion (BIC)** as score function.

a) The Bayesian Information Criterion (BIC) score

Let $\Gamma=(V, E)$ be a DAG, with $V=\{X_1,\ldots,X_n\}$ discrete r.v.

Let $x=(x_1,\ldots,x_n)$ be a data vector corresponding to r.v. $X_1,\ldots,X_n$, and let $D=\{\mathbf{x}_1,\ldots,\mathbf{x}_M\}$ be a set of M different data vectors (*"cases"*), that is, the *training data* T.

Let θ be the parameters associated to the model, that is, to the DAG. We are interested now in the notation to reflect its dependence on Γ, and then write θ^Γ .

We denote by d the "dimension" of Γ, that is, the number of non-redundant parameters in θ^Γ (d is a measure of the complexity of the model which is the BN with DAG Γ).

Remember that the likelihood of D being generated from model θ^{Γ} is the product of the probabilities of each case $\mathbf{x}_i$, given the model, that is:

$$L^D(\theta^{\Gamma}) = P(D/\theta^{\Gamma}) = \prod_{i=1}^{M} P(\mathbf{x}_i/\theta^{\Gamma})$$

$$= \prod_{i=1}^{M} P(X_1^i = x_1^i, \ldots, X_n^i = x_n^i/\theta\Gamma)$$

where $\mathbf{x}_i = (x_1^i, \ldots, x_n^i)$ is the $i-$th case of D

and $(X_1^i, \ldots, X_n^i)$ is a copy of $(X_1, \ldots, X_n)$.

Let θ^{Γ}_{MLE} be the Maximum Likelihood Estimation of θ^{Γ}, that is the value that is most likely to have generated the data set D.

The likelihood function reflects how well the model predicts the data that we indeed observe when we use the ML estimations as values for the parameters of the model.

Proposition (Corollary 12 Darwiche 2009):

If we obtain a DAG Γ^* from a DAG Γ by adding some directed arcs, then

$$L^D(\Theta^{\Gamma^*}{}_{MLE}) \geq L^D(\Theta^{\Gamma}{}_{MLE})$$

- Consequence: If we simply look for a network structure that maximizes the likelihood function, we would end up with a completely connected network (the maximum connected when the introduction of one more directed arc could not be possible without introducing cycles).
- Problems:
  - This would be a model too complex that can not be interpreted correctly (*Occam's razor principle*: among competing two models with similar predictions, the simpler one is the better).
  - It has many parameters to estimate, which can be excessive compared to the available data.
  - And even in the case that we have enough data, we will have *overfitting*: bad behaviour in cases that are not part of the data.

Solution:

Use the **Bayesian Information Criterion (BIC)** score, which is defined by:

$$BIC(\Gamma, D) = \ln\left(L^D(\theta^\Gamma_{MLE})\right) - \frac{d}{2}\ln(M)$$

where θ^Γ_{MLE} is the Maximum Likelihood Estimation of θ^Γ, $L^D(\theta^\Gamma_{MLE})$ is the value of the Likelihood function evaluated in θ^Γ_{MLE} , M is the number of cases in the data set D, d is the dimension of Γ, and ln( ) denotes the natural logarithm.

(The BIC score is based on Schwarz Information Criterion
G. Schwarz, "Estimating the dimension of a model". *Annals of Statistics*, 6:461–464, 1978)

$$BIC(\Gamma, D) = \ln\left(L^{D}(\theta^{\Gamma}_{MLE})\right) - \frac{d}{2}\ln(M)$$

The BIC score is intuitively appealing because it contains:

- ✓ the log-likelihood, which is a term that shows how well the model predicts the data when the parameter set is equal to its MLE value, and

- ✓ a term that punishes for model complexity.

The AIC (Akaike Information Criterion) score is similar:

$$AIC(\Gamma, D) = \ln\left(L^D(\theta^{\Gamma}_{MLE})\right) - d$$

(H. Akaike, “A new look at the statistical model identification”. *IEEE Transactions on Automatic Control*, 19:716–723, 1974)

AIC is a score that penalizes for complexity less than BIC if N is big enough (indeed, if N>7, since
ln(7)= 1.94591<2, while
ln(8)=2.079442 >2.

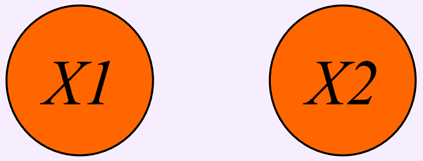
Therefore, the use of AIC as score function gives rise to DAGs **more connected** than that obtained using the BIC score.

An example

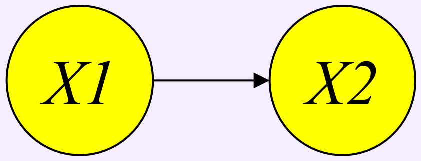
Suppose we have only two variables X_1 and X_2, each one taking only two values: 0 and 1.

In this case, as we have very few variables, we can exhaustively score all possible DAGs, which are three.

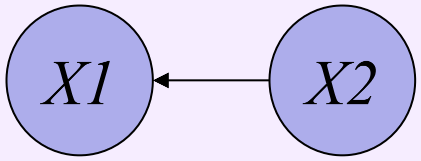
Then we will select the DAG(s) with the highest score.



DAG Γ_1

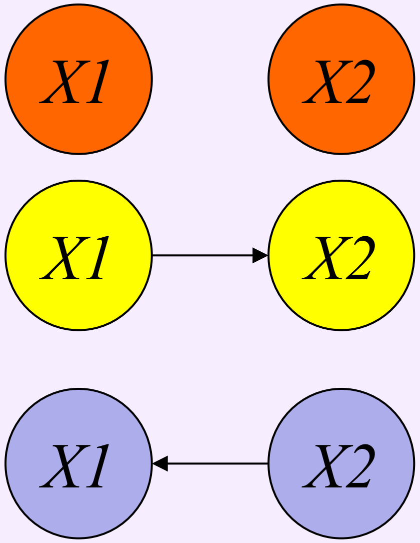


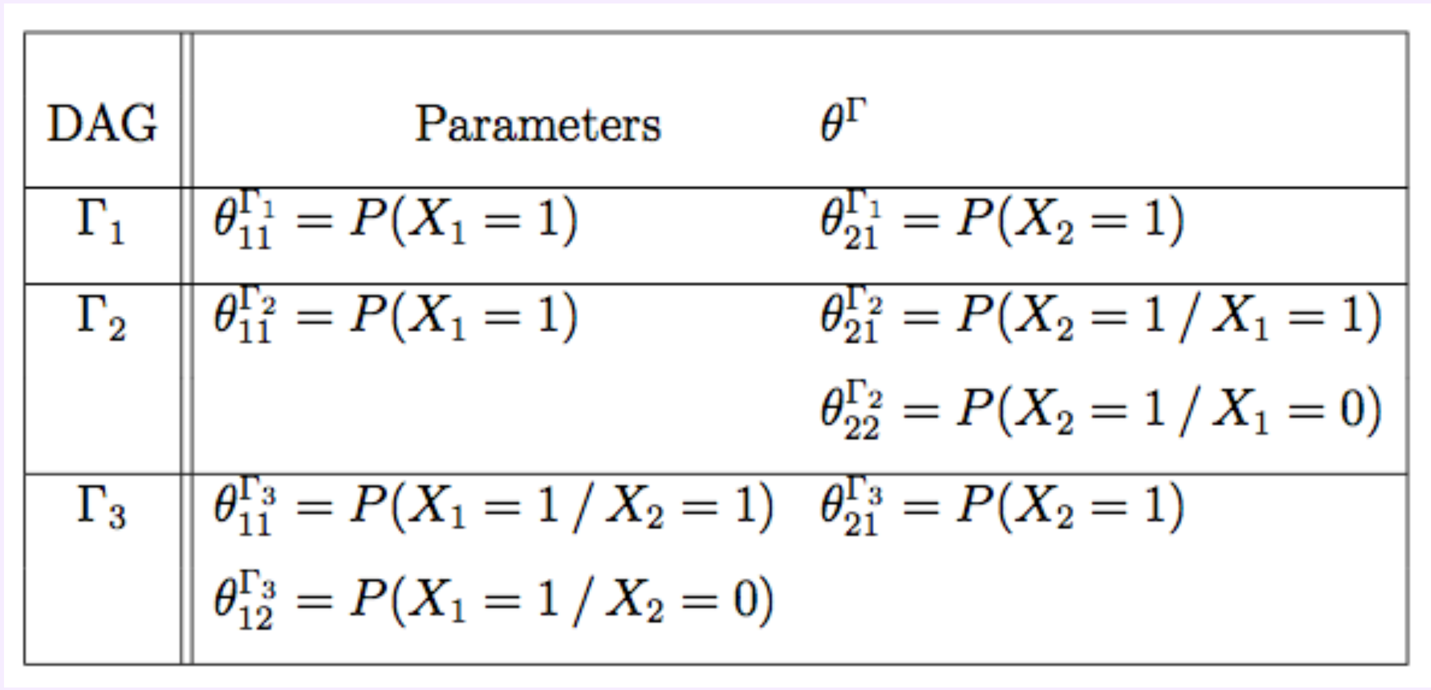
DAG Γ_2



DAG Γ_3

Which are the parameters of each DAG?



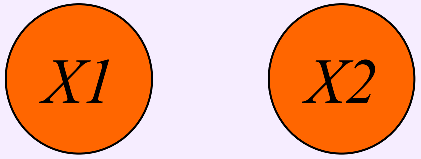


DAG	Parameters	θ^{Γ}
Γ_1	$\theta_{11}^{\Gamma_1} = P(X_1 = 1)$	$\theta_{21}^{\Gamma_1} = P(X_2 = 1)$
Γ_2	$\theta_{11}^{\Gamma_2} = P(X_1 = 1)$	$\theta_{21}^{\Gamma_2} = P(X_2 = 1 / X_1 = 1)$
		$\theta_{22}^{\Gamma_2} = P(X_2 = 1 / X_1 = 0)$
Γ_3	$\theta_{11}^{\Gamma_3} = P(X_1 = 1 / X_2 = 1)$	$\theta_{21}^{\Gamma_3} = P(X_2 = 1)$
	$\theta_{12}^{\Gamma_3} = P(X_1 = 1 / X_2 = 0)$	

Now we will compute BIC(Γ_1, D), BIC(Γ_2, D) and BIC(Γ_3, D) and choose the DAG(s) with highest value, where D is a the set of *training data* T.

Suppose we have the data D in the following table (with M=8):

Case	*X1*	*X2*
1	1	1
2	1	1
3	1	1
4	1	1
5	1	0
6	0	1
7	0	1
8	0	0



DAG Γ_1

For the DAG Γ_1, d=2, and the likelihood function is:

$$\begin{aligned}
L^D(\theta^{\Gamma_1}) &= P\big((X_1^1, X_2^1) = (x_1^1, x_2^1), \ldots, (X_1^M, X_2^M) = (x_1^M, x_2^M)/\theta^{\Gamma_1}\big) \\
&= \prod_{i=1}^{M} P((X_1^i, X_2^i) = (x_1^i, x_2^i)/\theta^{\Gamma_1}) \\
&= \prod_{i=1}^{M} P(X_1^i = x_1^i/\theta^{\Gamma_1})\, P(X_2^i = x_2^i/\theta^{\Gamma_1}) \\
&= \prod_{i=1}^{M} (\theta_{11}^{\Gamma_1})^{x_1^i}\,(1-\theta_{11}^{\Gamma_1})^{1-x_1^i}\,(\theta_{21}^{\Gamma_1})^{x_2^i}\,(1-\theta_{21}^{\Gamma_1})^{(1-x_2^i)} \\
&= (\theta_{11}^{\Gamma_1})^{s_{11}}\,(1-\theta_{11}^{\Gamma_1})^{t_{11}}\,(\theta_{21}^{\Gamma_1})^{s_{21}}\,(1-\theta_{21}^{\Gamma_1})^{t_{21}}
\end{aligned}$$

That is, $L^D(\theta^{\Gamma_1}) = (\theta_{11}^{\Gamma_1})^{s_{11}}\,(1-\theta_{11}^{\Gamma_1})^{t_{11}}\,(\theta_{21}^{\Gamma_1})^{s_{21}}\,(1-\theta_{21}^{\Gamma_1})^{t_{21}}$

Where

$\theta^{\Gamma_1} = (\theta_{11}^{\Gamma_1}, \theta_{21}^{\Gamma_1})$

s_{11} = number of cases in D for which $X_1 = 1$

t_{11} = number of cases in D for which $X_1 = 0$

s_{21} = number of cases in D for which $X_2 = 1$

t_{21} = number of cases in D for which $X_2 = 0$

Note that $s_{11} + t_{11} = s_{21} + t_{21} = M$

$$\ln(L^D(\theta^{\Gamma_1})) = s_{11}\,\ln(\theta_{11}^{\Gamma_1}) + t_{11}\,\ln(1-\theta_{11}^{\Gamma_1}) + s_{21}\,\ln(\theta_{21}^{\Gamma_1}) + t_{21}\,\ln(1-\theta_{21}^{\Gamma_1})$$

$$\frac{\partial L^D(\theta^{\Gamma_1})}{\partial\theta_{11}^{\Gamma_1}} = \frac{s_{11}}{\theta_{11}^{\Gamma_1}} - \frac{t_{11}}{1-\theta_{11}^{\Gamma_1}} = 0 \Rightarrow \theta_{11,MLE}^{\Gamma_1} = \frac{s_{11}}{s_{11}+t_{11}} = \frac{s_{11}}{M}$$

$$\frac{\partial L^D(\theta^{\Gamma_1})}{\partial\theta_{21}^{\Gamma_1}} = \frac{s_{21}}{\theta_{21}^{\Gamma_1}} - \frac{t_{21}}{1-\theta_{21}^{\Gamma_1}} = 0 \Rightarrow \theta_{21,MLE}^{\Gamma_1} = \frac{s_{21}}{s_{21}+t_{21}} = \frac{s_{21}}{M}$$

The Hessian matrix is a negative definite matrix for all the values of $\theta^{\Gamma 1}_{11}$ and $\theta^{\Gamma 1}_{21}$. Then, $L^D(\theta^{\Gamma 1})$ attains a local maximum (which indeed is a global maximum) at

$$\theta^{\Gamma_1}_{11,MLE} = \frac{s_{11}}{s_{11}+t_{11}} = \frac{s_{11}}{M}$$

$$\theta^{\Gamma_1}_{21,MLE} = \frac{s_{21}}{s_{21}+t_{21}} = \frac{s_{21}}{M}$$

Case	*X1*	*X2*
1	1	1
2	1	1
3	1	1
4	1	1
5	1	0
6	0	1
7	0	1
8	0	0

s11=5, t11=3, s21=6, t21=2

$$\theta^{\Gamma_1}_{MLE} = \left(\frac{s_{11}}{M}, \frac{s_{21}}{M}\right) = \left(\frac{5}{8}, \frac{6}{8}\right)$$

Then,

$$L^D(\theta^{\Gamma_1}_{MLE}) = (\theta^{\Gamma_1}_{11,MLE})^{s_{11}}\,(1-\theta^{\Gamma_1}_{11,MLE})^{t_{11}}\,(\theta^{\Gamma_1}_{21,MLE})^{s_{21}}\,(1-\theta^{\Gamma_1}_{21,MLE})^{t_{21}}$$
$$= \left(\frac{5}{8}\right)^5\left(\frac{3}{8}\right)^3\left(\frac{6}{8}\right)^6\left(\frac{2}{8}\right)^2 = 5.5594245067\times 10^{-5}$$

and the **Bayesian Information Criterion (BIC)** score for the DAG Γ1 is:

$$BIC(\Gamma_1, D) = \ln\left(L^D(\theta^{\Gamma_1}_{MLE})\right) - \frac{d}{2}\ln(M)$$
$$= -9.791187062 - \frac{2}{2}\ln(8) = -11.8706286$$

$$\begin{tikzcd} X1 \arrow[r] & X2 \end{tikzcd}$$

DAG Γ_2

For the DAG Γ_2, d=3, and the likelihood function is:

$$\begin{aligned}
L^D(\theta^{\Gamma_2}) &= P\big((X_1^1, X_2^1) = (x_1^1, x_2^1), \ldots, (X_1^M, X_2^M) = (x_1^M, x_2^M)/\theta^{\Gamma_2}\big) \\
&= \prod_{i=1}^{M} P((X_1^i, X_2^i) = (x_1^i, x_2^i)/\theta^{\Gamma_2}) \\
&= \prod_{i=1}^{M} P(X_1^i = x_1^i/\theta^{\Gamma_2})\, P(X_2^i = x_2^i/X_1^i = x_1^i, \theta^{\Gamma_2}) \\
&= \prod_{i=1}^{M} (\theta_{11}^{\Gamma_2})^{x_1^i} (1-\theta_{11}^{\Gamma_2})^{1-x_1^i} (\theta_{21}^{\Gamma_2})^{x_2^i x_1^i} (1-\theta_{21}^{\Gamma_2})^{(1-x_2^i)x_1^i} \\
&\qquad (\theta_{22}^{\Gamma_2})^{x_2^i (1-x_1^i)} (1-\theta_{22}^{\Gamma_2})^{(1-x_2^i)(1-x_1^i)} \\
&= (\theta_{11}^{\Gamma_2})^{s_{11}} (1-\theta_{11}^{\Gamma_2})^{t_{11}} (\theta_{21}^{\Gamma_2})^{s_{21}} (1-\theta_{21}^{\Gamma_2})^{t_{21}} (\theta_{22}^{\Gamma_2})^{s_{22}} (1-\theta_{22}^{\Gamma_2})^{t_{22}}
\end{aligned}$$

Where

$$\theta^{\Gamma_2} = (\theta^{\Gamma_2}_{11}, \theta^{\Gamma_2}_{21}, \theta^{\Gamma_2}_{22})$$

s_{11} = number of cases in D for which $X_1 = 1$

t_{11} = number of cases in D for which $X_1 = 0$

s_{21} = number of cases in D for which $X_2 = 1$ and $X1 = 1$

t_{21} = number of cases in D for which $X_2 = 0$ and $X1 = 1$

s_{22} = number of cases in D for which $X_2 = 1$ and $X1 = 0$

t_{22} = number of cases in D for which $X_2 = 0$ and $X1 = 0$

Note that $s_{11} + t_{11} = M$, $s_{21} + t_{21} = s_{11}$, $s_{22} + t_{22} = t_{11}$

$$\begin{aligned}\ln(L^D(\theta^{\Gamma_2})) &= s_{11}\ \ln(\theta^{\Gamma_2}_{11}) + t_{11}\ \ln(1-\theta^{\Gamma_2}_{11}) + s_{21}\ \ln(\theta^{\Gamma_2}_{21}) + t_{21}\ \ln(1-\theta^{\Gamma_2}_{21}) \\ &+ s_{22}\ \ln(\theta^{\Gamma_2}_{22}) + t_{22}\ \ln(1-\theta^{\Gamma_2}_{22})\end{aligned}$$

$$\frac{\partial L^D(\theta^{\Gamma_2})}{\partial\theta^{\Gamma_2}_{11}} = \frac{s_{11}}{\theta^{\Gamma_2}_{11}} - \frac{t_{11}}{1-\theta^{\Gamma_2}_{11}} = 0 \Rightarrow \theta^{\Gamma_2}_{11,MLE} = \frac{s_{11}}{s_{11}+t_{11}} = \frac{s_{11}}{M}$$

$$\frac{\partial L^D(\theta^{\Gamma_2})}{\partial\theta^{\Gamma_2}_{21}} = \frac{s_{21}}{\theta^{\Gamma_2}_{21}} - \frac{t_{21}}{1-\theta^{\Gamma_2}_{21}} = 0 \Rightarrow \theta^{\Gamma_2}_{21,MLE} = \frac{s_{21}}{s_{21}+t_{21}} = \frac{s_{21}}{s_{11}}$$

$$\frac{\partial L^D(\theta^{\Gamma_2})}{\partial\theta^{\Gamma_2}_{22}} = \frac{s_{22}}{\theta^{\Gamma_2}_{22}} - \frac{t_{22}}{1-\theta^{\Gamma_2}_{22}} = 0 \Rightarrow \theta^{\Gamma_2}_{22,MLE} = \frac{s_{22}}{s_{22}+t_{22}} = \frac{s_{22}}{t_{11}}$$

We can check that Hessian matrix is a negative definite matrix for all the values of $\theta^{\Gamma 2}_{11}$, $\theta^{\Gamma 2}_{21}$ and $\theta^{\Gamma 2}_{22}$. Then, $L^D(\theta^{\Gamma 2})$ attains a local maximum (which is indeed a global maximum) at

$$\theta^{\Gamma_2}_{11,MLE} = \frac{s_{11}}{s_{11}+t_{11}} = \frac{s_{11}}{M}$$

$$\theta^{\Gamma_2}_{21,MLE} = \frac{s_{21}}{s_{21}+t_{21}} = \frac{s_{21}}{s_{11}}$$

$$\theta^{\Gamma_2}_{22,MLE} = \frac{s_{22}}{s_{22}+t_{22}} = \frac{s_{22}}{t_{11}}$$

Case	*X1*	*X2*
1	1	1
2	1	1
3	1	1
4	1	1
5	1	0
6	0	1
7	0	1
8	0	0

$\Rightarrow$

s11=5, t11=3,
s21=4, t21=1,
s22=2, t22=1

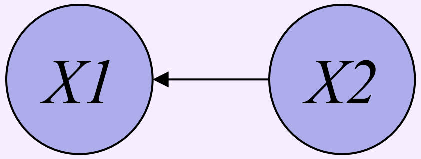
$$\theta^{\Gamma_2}_{MLE} = \left(\frac{s_{11}}{M}, \frac{s_{21}}{s_{11}}, \frac{s_{22}}{t_{11}}\right) = \left(\frac{5}{8}, \frac{4}{5}, \frac{2}{3}\right)$$

Then,

$$L^D(\theta^{\Gamma_2}) = (\theta_{11}^{\Gamma_2})^{s_{11}} (1-\theta_{11}^{\Gamma_2})^{t_{11}} (\theta_{21}^{\Gamma_2})^{s_{21}} (1-\theta_{21}^{\Gamma_2})^{t_{21}} (\theta_{22}^{\Gamma_2})^{s_{22}} (1-\theta_{22}^{\Gamma_2})^{t_{22}}$$
$$= \left(\frac{5}{8}\right)^5 \left(\frac{3}{8}\right)^3 \left(\frac{4}{5}\right)^4 \left(\frac{1}{5}\right)^1 \left(\frac{2}{3}\right)^2 \left(\frac{1}{3}\right)^1 = 6.103515625 \times 10^{-5}$$

and the **Bayesian Information Criterion (BIC)** score for the DAG Γ2 is:

$$BIC(\Gamma_2, D) = \ln\left(L^D(\theta_{MLE}^{\Gamma_2})\right) - \frac{d}{2}\ln(M)$$
$$= -9.704060528 - \frac{3}{2}\ln(8) = -12.82322284$$



DAG Γ_3

For the DAG Γ_3, d=3, and the likelihood function is:

$$
\begin{aligned}
L^D(\theta^{\Gamma_3}) &= P\big((X_1^1, X_2^1) = (x_1^1, x_2^1), \ldots, (X_1^M, X_2^M) = (x_1^M, x_2^M)/\theta^{\Gamma_3}\big) \\
&= \prod_{i=1}^{M} P((X_1^i, X_2^i) = (x_1^i, x_2^i)/\theta^{\Gamma_3}) \\
&= \prod_{i=1}^{M} P(X_1^i = x_1^i/X_2^i = x_2^i, \theta^{\Gamma_3})\, P(X_2^i = x_2^i/\theta^{\Gamma_3}) \\
&= \prod_{i=1}^{M} (\theta_{11}^{\Gamma_3})^{x_1^i x_2^i} (1-\theta_{11}^{\Gamma_3})^{(1-x_1^i)x_2^i} (\theta_{12}^{\Gamma_3})^{x_1^i(1-x_2^i)} (1-\theta_{12}^{\Gamma_3})^{(1-x_1^i)(1-x_2^i)} \\
&\qquad (\theta_{21}^{\Gamma_3})^{x_2^i} (1-\theta_{21}^{\Gamma_3})^{(1-x_2^i)} \\
&= (\theta_{11}^{\Gamma_3})^{s_{11}} (1-\theta_{11}^{\Gamma_3})^{t_{11}} (\theta_{12}^{\Gamma_3})^{s_{12}} (1-\theta_{12}^{\Gamma_3})^{t_{12}} (\theta_{21}^{\Gamma_3})^{s_{21}} (1-\theta_{21}^{\Gamma_3})^{t_{21}}
\end{aligned}
$$

Where

$$\theta^{\Gamma_3} = (\theta_{11}^{\Gamma_3}, \theta_{12}^{\Gamma_3}, \theta_{21}^{\Gamma_3})$$

s_{11} = number of cases in D for which $X_1 = 1$ and $X2 = 1$

t_{11} = number of cases in D for which $X_1 = 0$ and $X2 = 1$

s_{12} = number of cases in D for which $X_1 = 1$ and $X2 = 0$

t_{12} = number of cases in D for which $X_1 = 0$ and $X2 = 0$

s_{21} = number of cases in D for which $X_2 = 1$

t_{21} = number of cases in D for which $X_2 = 0$

Note that $s_{11} + t_{11} = s_{21}$, $s_{12} + t_{12} = t_{21}$, $s_{21} + t_{21} = M$

$$\ln(L^D(\theta^{\Gamma_3})) = s_{11}\ln(\theta_{11}^{\Gamma_3}) + t_{11}\ln(1-\theta_{11}^{\Gamma_3}) + s_{12}\ln(\theta_{12}^{\Gamma_3}) + t_{12}\ln(1-\theta_{12}^{\Gamma_3}) + s_{21}\ln(\theta_{21}^{\Gamma_3}) + t_{21}\ln(1-\theta_{21}^{\Gamma_3})$$

$$\frac{\partial L^D(\theta^{\Gamma_3})}{\partial\theta_{11}^{\Gamma_3}} = \frac{s_{11}}{\theta_{11}^{\Gamma_3}} - \frac{t_{11}}{1-\theta_{11}^{\Gamma_3}} = 0 \Rightarrow \theta_{11,MLE}^{\Gamma_3} = \frac{s_{11}}{s_{11}+t_{11}} = \frac{s_{11}}{s_{21}}$$

$$\frac{\partial L^D(\theta^{\Gamma_3})}{\partial\theta_{12}^{\Gamma_3}} = \frac{s_{12}}{\theta_{12}^{\Gamma_3}} - \frac{t_{12}}{1-\theta_{12}^{\Gamma_3}} = 0 \Rightarrow \theta_{12,MLE}^{\Gamma_3} = \frac{s_{12}}{s_{12}+t_{12}} = \frac{s_{12}}{t_{21}}$$

$$\frac{\partial L^D(\theta^{\Gamma_3})}{\partial\theta_{21}^{\Gamma_3}} = \frac{s_{21}}{\theta_{21}^{\Gamma_3}} - \frac{t_{21}}{1-\theta_{21}^{\Gamma_3}} = 0 \Rightarrow \theta_{21,MLE}^{\Gamma_3} = \frac{s_{21}}{s_{21}+t_{21}} = \frac{s_{21}}{M}$$

We can check that Hessian matrix is a negative definite matrix for all the values of $\theta^{\Gamma 3}{}_{11}$, $\theta^{\Gamma 3}{}_{12}$ and $\theta^{\Gamma 3}{}_{21}$. Then, $L^D(\theta^{\Gamma 3})$ attains a local maximum (which is indeed a global maximum) at

$$\theta^{\Gamma_2}_{11,MLE} = \frac{s_{11}}{s_{11}+t_{11}} = \frac{s_{11}}{M}$$

$$\theta^{\Gamma_2}_{21,MLE} = \frac{s_{21}}{s_{21}+t_{21}} = \frac{s_{21}}{s_{11}}$$

$$\theta^{\Gamma_2}_{22,MLE} = \frac{s_{22}}{s_{22}+t_{22}} = \frac{s_{22}}{t_{11}}$$

Case	*X1*	*X2*
1	1	1
2	1	1
3	1	1
4	1	1
5	1	0
6	0	1
7	0	1
8	0	0

$\Rightarrow$

s11=4, t11=2,
s12=1, t12=1,
s21=6, t21=2

$$\theta^{\Gamma_3}_{MLE} = \left(\frac{s_{11}}{s_{21}}, \frac{s_{12}}{t_{21}}, \frac{s_{21}}{M}\right) = \left(\frac{4}{6}, \frac{1}{2}, \frac{6}{8}\right)$$

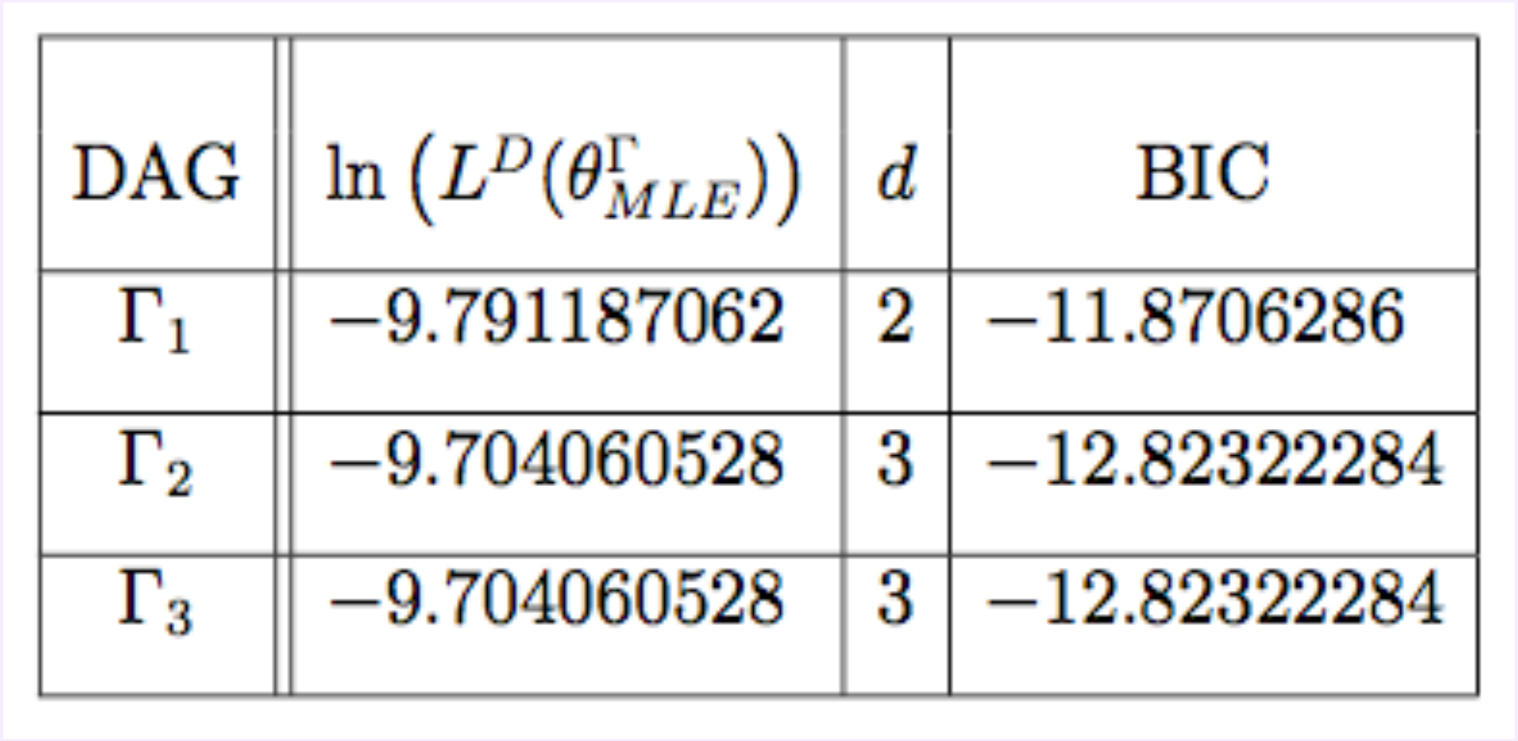
Then,

$$L^D(\theta^{\Gamma_3}) = (\theta_{11}^{\Gamma_3})^{s_{11}}\,(1-\theta_{11}^{\Gamma_3})^{t_{11}}\,(\theta_{12}^{\Gamma_3})^{s_{12}}\,(1-\theta_{12}^{\Gamma_3})^{t_{12}}\,(\theta_{21}^{\Gamma_3})^{s_{21}}\,(1-\theta_{21}^{\Gamma_3})^{t_{21}}$$
$$= \left(\frac{4}{6}\right)^4\left(\frac{2}{6}\right)^2\left(\frac{1}{2}\right)^1\left(\frac{1}{2}\right)^1\left(\frac{6}{8}\right)^6\left(\frac{2}{8}\right)^2 = 6.103515625\times 10^{-5}$$

and the **Bayesian Information Criterion (BIC)** score for the DAG Γ3 is:

$$BIC(\Gamma_3, D) = \ln\left(L^D(\theta_{MLE}^{\Gamma_3})\right) - \frac{d}{2}\ln(M)$$
$$= -9.704060528 - \frac{3}{2}\ln(8) = -12.82322284$$

Observe that although the data were more probable given Γ_2 or Γ_3, Γ_1 won with the **Bayesian Information Criterion (BIC)** score criterion because it is less complex.



DAG	$\ln\left(L^D(\theta^\Gamma_{MLE})\right)$	d	BIC
Γ_1	-9.791187062	2	-11.8706286
Γ_2	-9.704060528	3	-12.82322284
Γ_3	-9.704060528	3	-12.82322284

We saw with the example that when there are not many variables, we can exhaustively score all possible DAGs and select the DAG(s) with the highest score.

However, when the number of variables is not small, it is a computationally unfeasible problem since the number of DAGs containing *n* Boolean nodes is given by the following recurrence, and researchers have been forced to develop heuristic DAG search algorithms!!

$$f(n) = \begin{cases} 1 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i) & \text{if } n \geq 2 \end{cases}$$

(Robinson, R.W., "Counting Unlabeled Acyclic Diagraphs", in C.H.C. Little (Ed.): *Lecture Notes in Mathematics, 622: Combinatorial Mathematics V*, Springer-Verlag, New York, 1977.)

The number of DAGs on n nodes, for n = 1, 2, 3, … is:
1, 3, 25, 543, 29281, 3781503, 1138779265, 783702329343, 1213442454842881, 4175098976430598143,…
(sequence A003024 in the OEIS, the On-Line Encyclopedia of Integer Sequences).

b) An heuristic DAG search algorithm: the Greedy Search-And-Score Algorithm

In this subsection we present an heuristic algorithm for model selection that search over DAGs, that is, in which the search space consists of DAGs.

Specifically, the search space is the set of all DAGs containing *n* nodes, where *n* is the number of random variables.

The goal is to find a DAG with the maximum score, where our scoring criterion could be the Bayesian Information Criterion (BIC) score (or some other score).

To fix ideas, we focus on the **BIC score**.

Recall that the BIC score is:

$$BIC(\Gamma, D) = \ln\left(L^D(\theta^{\Gamma}_{MLE})\right) - \frac{d}{2}\ln(M)$$

The log-likelihood can be expressed in the following way:
(R. R. Bouckaert, “Belief networks construction using the minimum description length principle”. *Lecture Notes in Computer Science*, 747:41–48, 1993)

$$\ln\left(L^D(\theta^{\Gamma}_{EML})\right) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\ln\left(\frac{N_{ijk}}{N_{ij}}\right)$$

and the dimension d in the penalizing term in this way:

$$d = \sum_{i=1}^{n}(r_i - 1)\, q_i$$

Then, $$BIC(\Gamma, D) = \sum_{i=1}^{n} g(\{X_i, PA_{X_i}\}, D)$$ with

$$g(\{X_i, PA_{X_i}\}, D) = \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \ln\left(\frac{N_{ijk}}{N_{ij}}\right) - \frac{(r_i - 1)\, q_i}{2} \ln(M)$$ where

n is the number of variables $\{X_1, \ldots, X_n\}$

r_i is the number of states of variable X_i, $i = 1, \ldots, n$

q_i is the number of possible configurations of the parent set PA_{X_i} on the DAG Γ

(therefore, $q_i = \prod_{j:X_j \in PA_{X_i}} r_j$)

ω_{ij} ($j = 1, \ldots, q_i$) represents a configuration of PA_{X_i}

N_{ijk} is the number of instances in the data set D where the variable X_i takes its k–th value and the set of variables PA_{X_i} have the configuration ω_{ij}

$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ is the number of instances in the data set D where the variables in PA_{X_i} have their j–th configuration ω_{ij}

That is, we have the following useful property:

Property: The BIC score is *decomposable*. This means that the total score can be computed as the sum of local scores of the nodes in the DAG:

$$BIC(\Gamma, D) = \sum_{i=1}^{n} g(\{X_i, PA_{X_i}\}, D)$$

Where the local score for each node X_i depends only on data values of X_i and nodes in PA_{X_i}:

$$g(\{X_i, PA_{X_i}\}, D) = \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \ln\left(\frac{N_{ijk}}{N_{ij}}\right) - \frac{(r_i - 1)\, q_i}{2} \ln(M)$$

This makes it easy to compute the effect of a simple modification to the network (such as adding or removing an edge).

When a model searching algorithm need only locally recompute a few scores to determine the score for the next model under consideration, we say the algorithm has **local scoring updating**. A model with local scoring updating is considerably more efficient than one without it.

The Greedy Search-And-Score Algorithm with the BIC score has local scoring updating and performs a search through the space of possible network structures. The operators used to perform this search, given a current candidate, are:

- ✓ add and arc,
- ✓ remove an arc, and
- ✓ reverse an arc,

And all operations are subject to the constraint that the resultant graph does not contain a cycle.

The Greedy Search-And-Score Algorithm based on the data set D

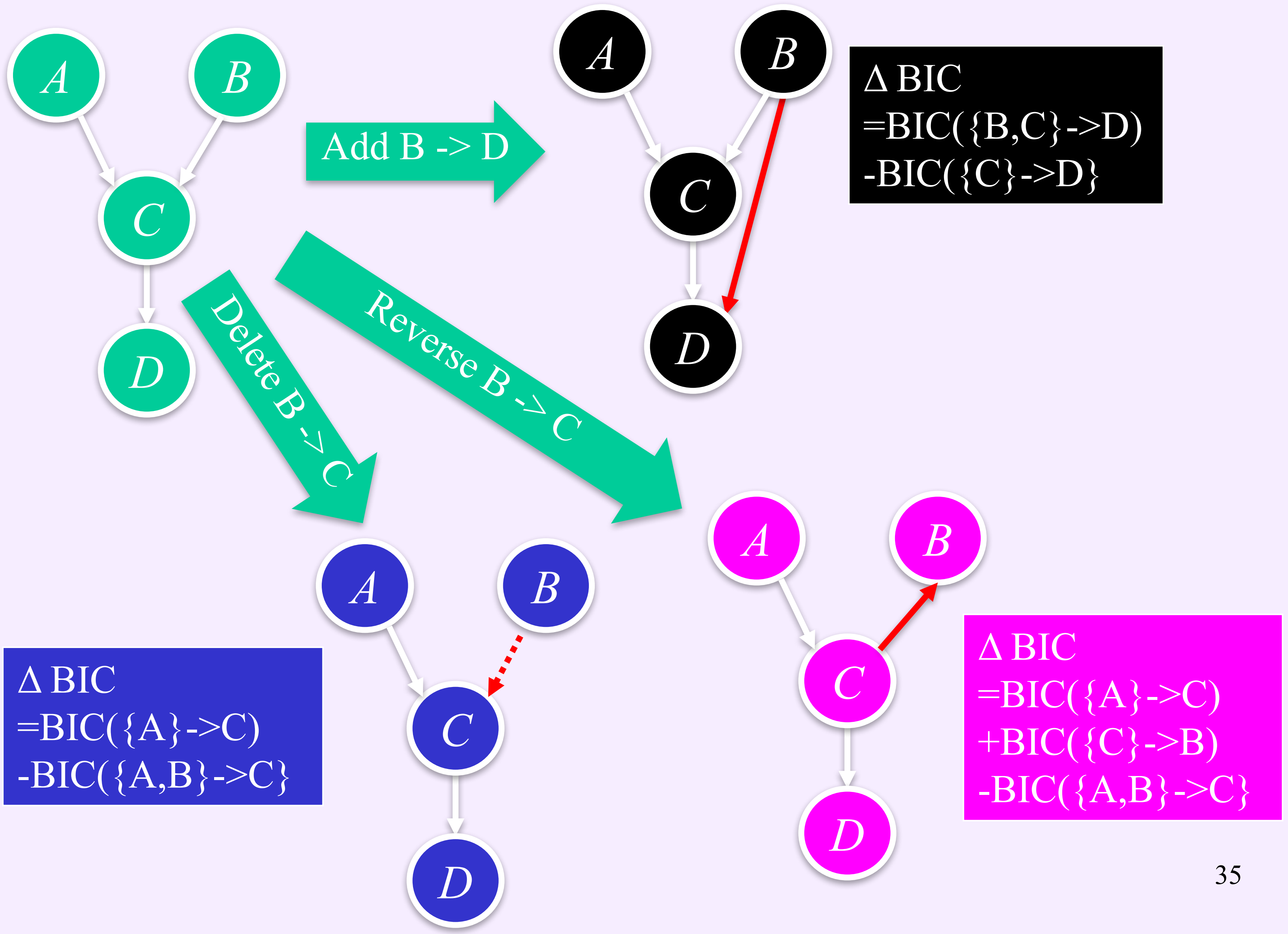
1. Start with the DAG with no edges (arrows).
2. At each step of the search, of all those DAGs in the neighbourhood of our current DAG (that is, the set of DAGs Γ' obtained from our current DAG Γ by applying one of the operators), we "greedily" choose the one that maximizes BIC(Γ', D).
3. We halt when no operation increases this score.

Note: in each step, if an edge to X_i is added or deleted from Γ, we need only re-evaluate g({X_i, PA_{Xi}}, D), where PA_{Xi} is now the set of parents of X_i in Γ'.
If an edge between X_i and X_j is reversed, we need only re-evaluate g({X_i, PA_{Xi}}, D) and g({X_j, PA_{Xj}}, D). Then, this algorithm has **local scoring updating**.

Local scoring updating: To evaluate the effect of a change to a DAG Γ that involves a certain X_i node and give rise to another DAG Γ', BIC (Γ', D) must be obtained from BIC (Γ, D), and to do this we only have to

- ✓ re-evaluate g({X_i, PA(X_i)}, D), if what we do is add or remove an arc that reaches X_i,
- ✓ re-evaluate g({X_i, PA(X_i)}, D) and g({X_j, PA(X_j)}, D) if we change the direction of an arc between X_i and X_j.

To update score after local change, only re-score families that changed!



Δ BIC
=BIC({B,C}->D)
-BIC({C}->D}

Δ BIC
=BIC({A}->C)
-BIC({A,B}->C}

Δ BIC
=BIC({A}->C)
+BIC({C}->B)
-BIC({A,B}->C}

Problem (with greedy search algorithm):
it could halt at a candidate solution that **locally** maximizes the objective function rather than **globally** maximizes it.
Pseudo-Solutions:

- **iterated hill-climbing algorithm**: Local search is done until a **local** maximum is obtained. Then, the current structure is randomly perturbed, and the process is repeated: from this new DAG a new search of a local maximum is started. The process is repeated several times. Finally, the maximum over local maxima is used.
- **random-restart hill-climbing algorithm**: It does the same thing than iterated hill-climbing algorithm but starts each time in a randomly chosen DAG.
- **Tabu search**: a modified ill-climbing that keeps a list of the last k structures visited, and returns only if they are all worse that the current one.
- **Simulated annealing**: It is another algorithm that addresses this issue but allowing the value of the score in some steps to decrease, temporarily worsening, and then improving again, in order to scape from the attraction of a local maximum.
- **Genetic algorithms**: they perturb (mutation) and combine crossover features through several generations of structures, and keep the ones leading to better scores. Inspired by Darwinian evolution rules.

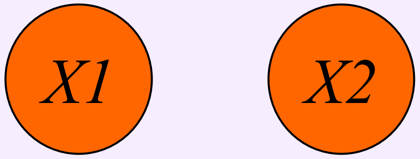
c) Score-Based Structure Learning with bnlearn

As for the case of parameters learning, there exist different ways of creating the structure of a BN (a bn object):

1. Expert-driven approach: in which the structure (bn object) is specified by the user, as did in Block 3. After that, we can learn parameters of this BN (that is, create a bn.fit object) as saw in Block 3.
2. Data-driven approach: learning structure from a data set using hc(),which allows to learn structure with bnlearn using the iterated hill-climbing (hc) algorithm, and creates a bn object.
3. Hybrid approach: combining the above (using the **black and white lists** in hc() to forbid and/or impose some arrows in the structure).

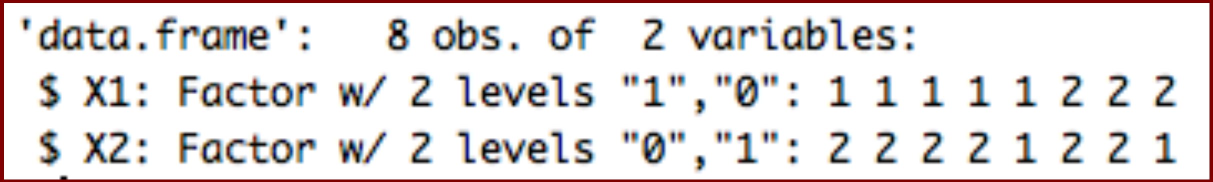
Example

In the previous example, we have two Boolean variables, and the "training" data set D. This data set must be introduced in R as a data frame, say of name "training".



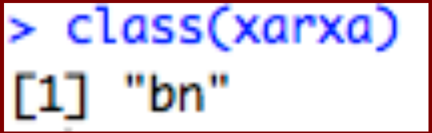
Case	*X1*	*X2*
1	1	1
2	1	1
3	1	1
4	1	1
5	1	0
6	0	1
7	0	1
8	0	0

```
> library(bnlearn)
> library(Rgraphviz)
> load("/Users/…/training.rdata")
> str(training)
```



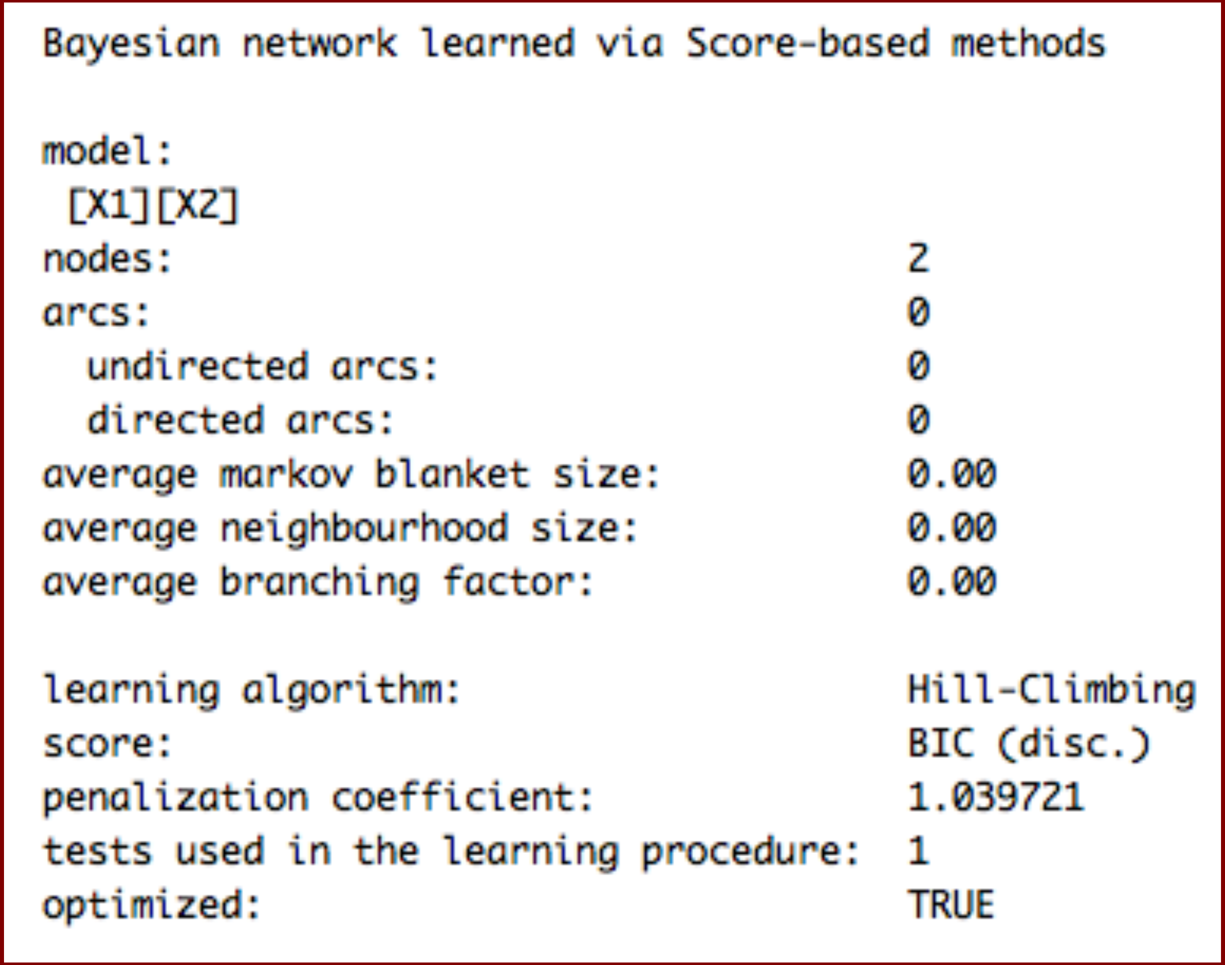
```
'data.frame':   8 obs. of  2 variables:
 $ X1: Factor w/ 2 levels "1","0": 1 1 1 1 1 2 2 2
 $ X2: Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 1
```

```
> xarxa<-hc(training, score="bic")
> class(xarxa)
```



```
> class(xarxa)
[1] "bn"
```

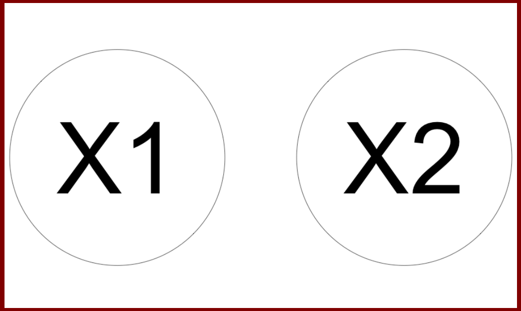
> xarxa



```
Bayesian network learned via Score-based methods

model:
 [X1][X2]
nodes:                                 2
arcs:                                  0
  undirected arcs:                     0
  directed arcs:                       0
average markov blanket size:           0.00
average neighbourhood size:            0.00
average branching factor:              0.00

learning algorithm:                    Hill-Climbing
score:                                 BIC (disc.)
penalization coefficient:              1.039721
tests used in the learning procedure:  1
optimized:                             TRUE
```

(the "penalization coefficient" is **ln(M)/2**)

```
> plot<-graphviz.plot(xarxa)
> plot
```

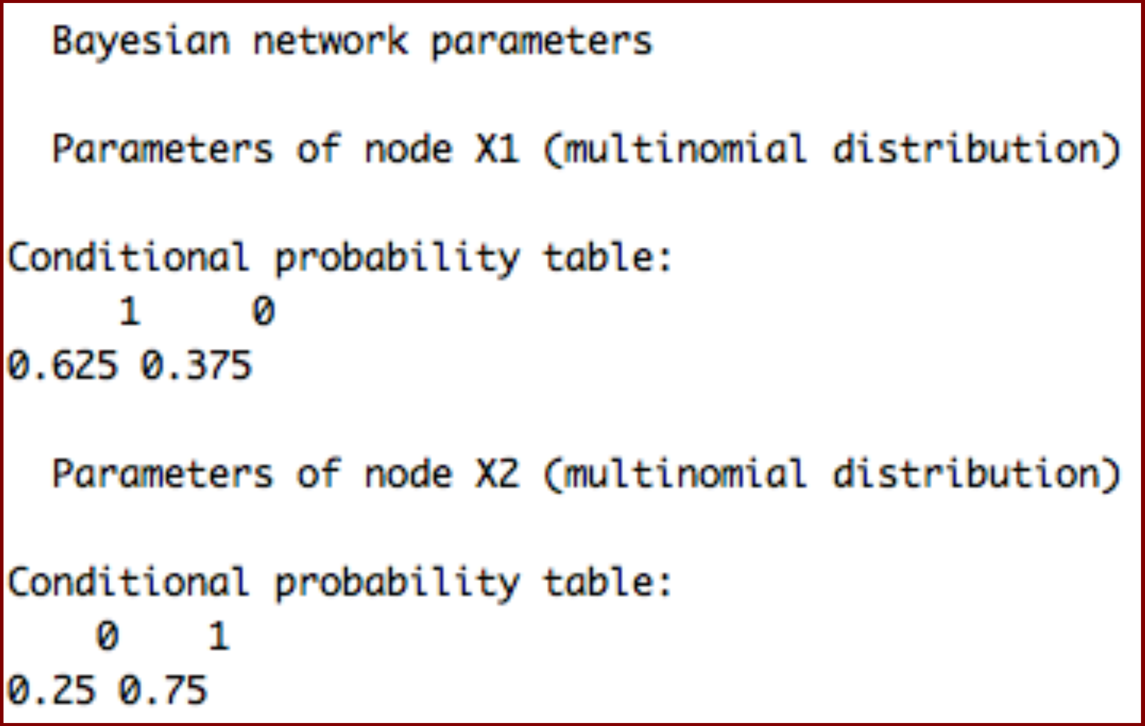


```
> logLik(xarxa,training)
[1] -9.791187
> BIC(xarxa,training)
[1] -11.87063
```

(compare with page 26: the “winner” is the same DAG, Γ_1, and the values of logLik and BIC are as found by hand)

Once we have the structure ("xarxa", which is a bn object), we can estimate parameters as in Block 3, obtaining a bn.fit object that we name "xarxa.estimada".

```
> xarxa.estimada=bn.fit(xarxa, training, method="mle")
> xarxa.estimada
```



```
  Bayesian network parameters

  Parameters of node X1 (multinomial distribution)

Conditional probability table:
    1     0
0.625 0.375

  Parameters of node X2 (multinomial distribution)

Conditional probability table:
   0    1
0.25 0.75
```

Remark: the Hybrid approach

This approach combines the Data-driven approach (learning structure exclusively from a data set) using the **black and white lists** in hc() to forbid and/or impose some arrows in the structure).

More specifically, we can use

whitelist=name1

inside function hc, where name1 is a data frame with two columns (optionally labeled "from" and "to") containing a set of arcs **to be included** necessarily in the graph, and/or

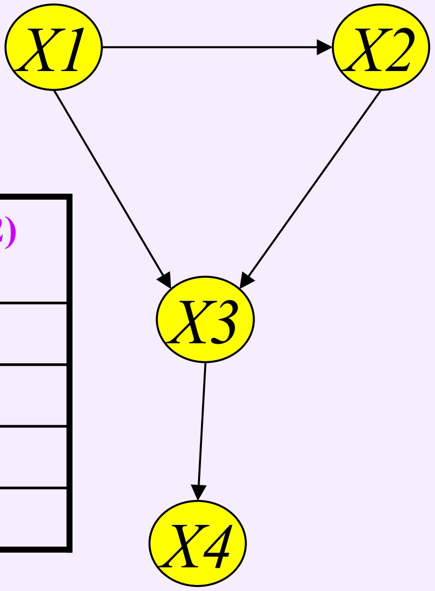
blacklist=name2

where name2 is a data frame with two columns (also optionally labeled "from" and "to") containing a set of arcs **not** to be included in the graph.

Another example

X1	P(X2=0 / X1)	P(X2=1 / X1)
0	0.3	0.70
1	0.8	0.20

P(X1=0)	P(X1=1)
0.80	0.20



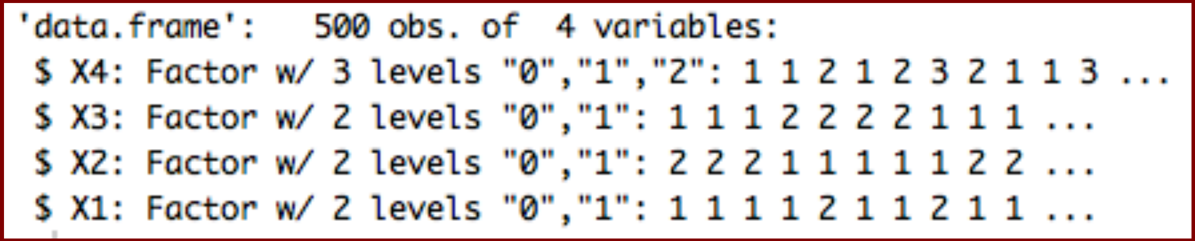
$$\begin{tikzcd} X1 \arrow[r] \arrow[dr] & X2 \arrow[d] \\ & X3 \arrow[d] \\ & X4 \end{tikzcd}$$

X1	X2	P(X3=0/X1,X2)	P(X3=1X1,X2)
0	0	0.10	0.90
0	1	0.80	0.20
1	0	0.50	0.50
1	1	0.30	0.70

X3	P(X4=0/X3)	P(X4=1/X3)	P(X4=2/X3)
0	0.60	0.30	0.10
1	0.20	0.30	0.50

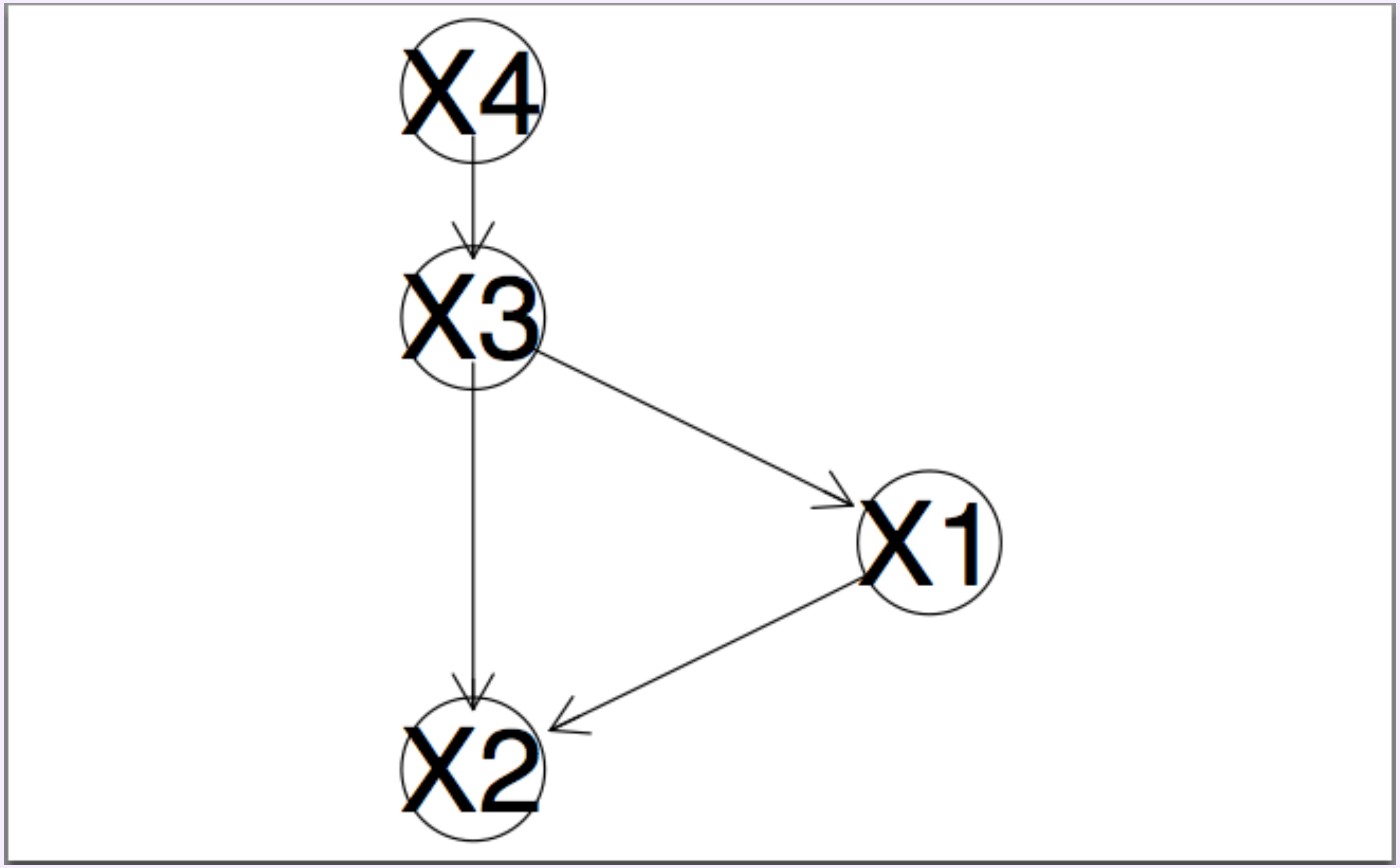
Load file “Exemple_dades_generades_Hugin.rdata”, which contains 500 cases randomly generated from the above Bayesian network. We will use it to learn the BN by using R with package bnlearn.

```
> library(bnlearn)
> library(Rgraphviz)
> load("/…/Exemple_dades_generades_Hugin.rdata")
> str(Exemple_dades_generades_Hugin)
```



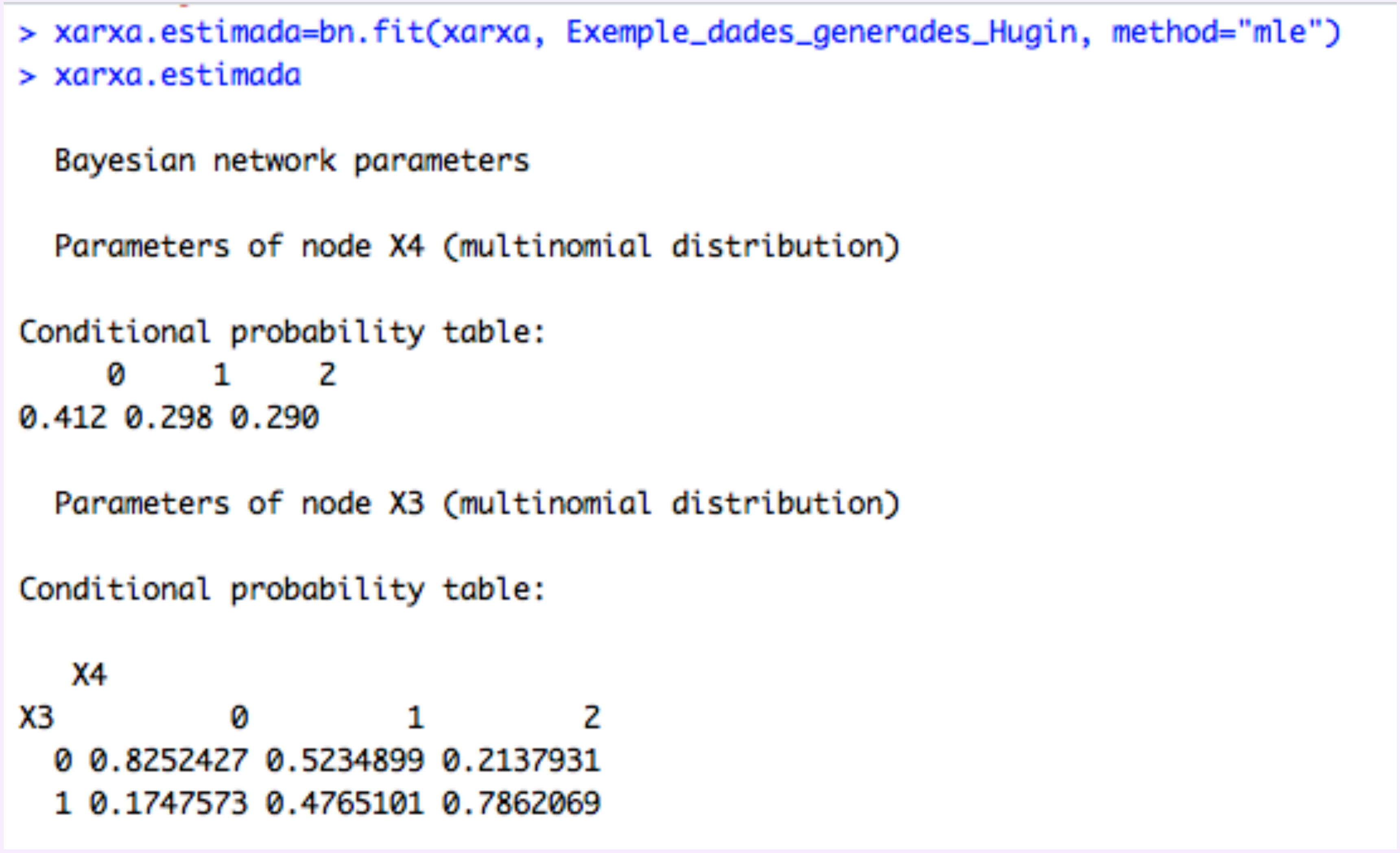
```
'data.frame':   500 obs. of  4 variables:
 $ X4: Factor w/ 3 levels "0","1","2": 1 1 2 1 2 3 2 1 1 3 ...
 $ X3: Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 1 1 1 ...
 $ X2: Factor w/ 2 levels "0","1": 2 2 2 1 1 1 1 1 2 2 ...
 $ X1: Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
```

```
> xarxa<-hc(Exemple_dades_generades_Hugin, score="bic”)
> class(xarxa)
> xarxa
> plot<-graphviz.plot(xarxa)
> plot
```

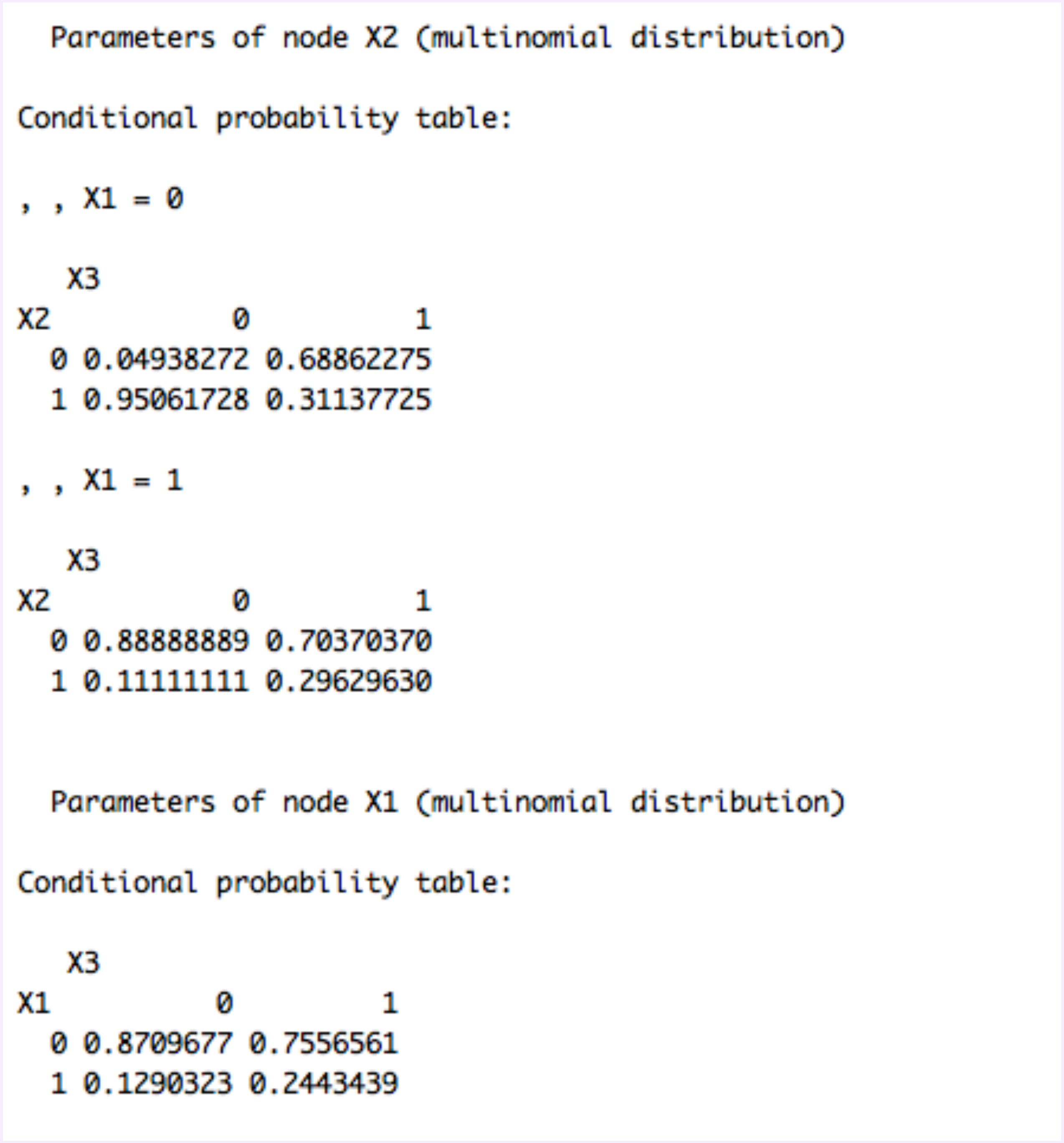


Compare with the structure we used to generate data. They are not exactly the same because some direction change.

Now we estimate the parameters with bnlearn:

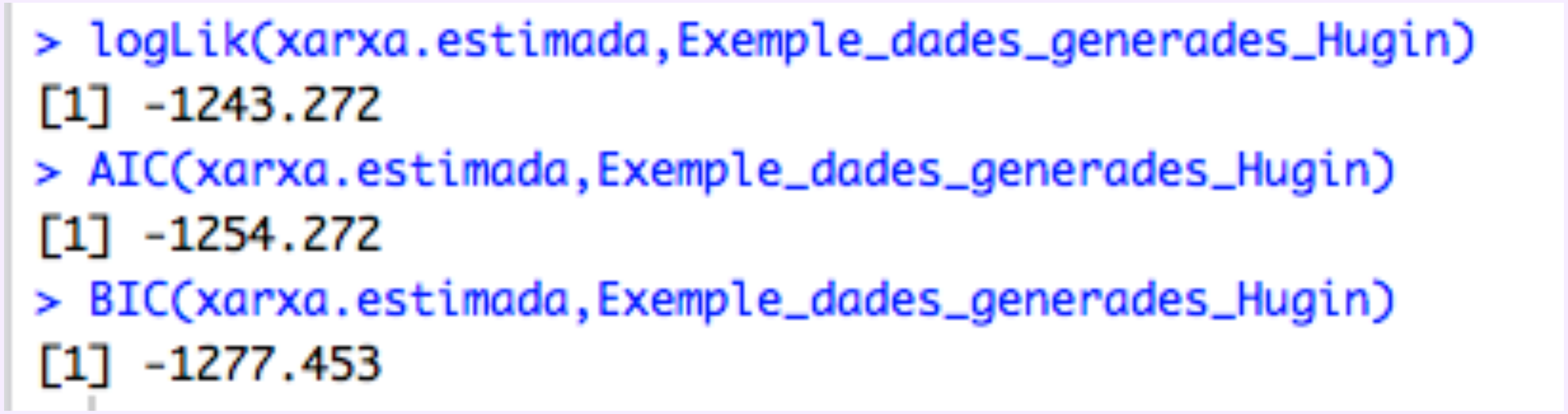


```
> xarxa.estimada=bn.fit(xarxa, Exemple_dades_generades_Hugin, method="mle")
> xarxa.estimada

  Bayesian network parameters

  Parameters of node X4 (multinomial distribution)

Conditional probability table:
    0     1     2
0.412 0.298 0.290

  Parameters of node X3 (multinomial distribution)

Conditional probability table:

   X4
X3          0         1         2
  0 0.8252427 0.5234899 0.2137931
  1 0.1747573 0.4765101 0.7862069
```



```
  Parameters of node X2 (multinomial distribution)

Conditional probability table:

, , X1 = 0

   X3
X2           0          1
  0 0.04938272 0.68862275
  1 0.95061728 0.31137725

, , X1 = 1

   X3
X2           0          1
  0 0.88888889 0.70370370
  1 0.11111111 0.29629630


  Parameters of node X1 (multinomial distribution)

Conditional probability table:

   X3
X1          0         1
  0 0.8709677 0.7556561
  1 0.1290323 0.2443439
```

Finally, we can compute the value of the score functions:
- ✓ Log-likelihood,
- ✓ AIC, and
- ✓ BIC

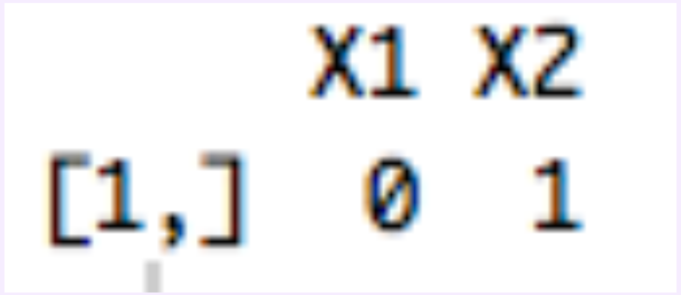


```
> logLik(xarxa.estimada,Exemple_dades_generades_Hugin)
[1] -1243.272
> AIC(xarxa.estimada,Exemple_dades_generades_Hugin)
[1] -1254.272
> BIC(xarxa.estimada,Exemple_dades_generades_Hugin)
[1] -1277.453
```

How to predict the value of **X4** from the evidence that **X1=0 and X2=1**, for example? We will use the package gRain for doing that with R:

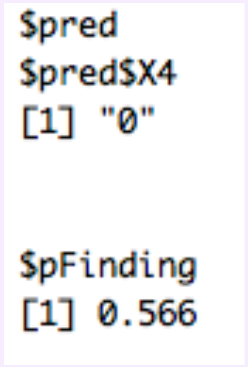
```
> library(gRain)
> xarxa.estimada.gRain<-as.grain(xarxa.estimada)
> #
> valores<-c(0,1)
> eevvii<-matrix(valores,nrow=1,byrow=T)
> colnames(eevvii)<-c("X1","X2")
> eevvii
```

eevvii is the evidence:



```
     X1 X2
[1,]  0  1
```

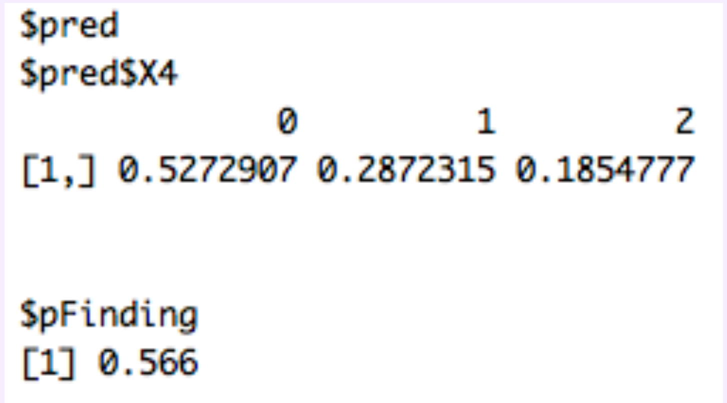
```
> evi<-as.data.frame(eevvii)
> pp<-predict(xarxa.estimada.gRain, response="X4", newdata=evi,
predictor=c("X1","X2"), type="class")
> pp
```



```
$pred
$pred$X4
[1] "0"


$pFinding
[1] 0.566
```

The prediction for X4 is “0”, and the probability of the evidence is P(X1=0, X2=1)=0.566

```
> ppdist<-predict(xarxa.estimada.gRain, response="X4",
newdata=evi, predictor=c("X1","X2"), type="distribution")
> ppdist
```

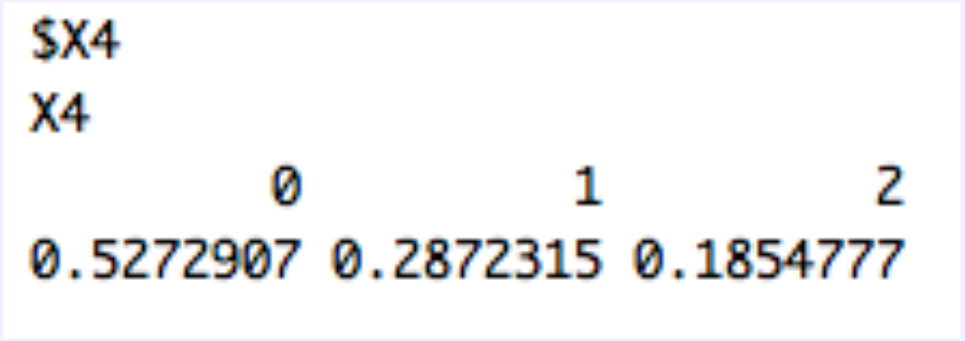


```
$pred
$pred$X4
             0         1         2
[1,] 0.5272907 0.2872315 0.1854777


$pFinding
[1] 0.566
```

The value “0”, which is the prediction for X4, is that with the maximum probability.

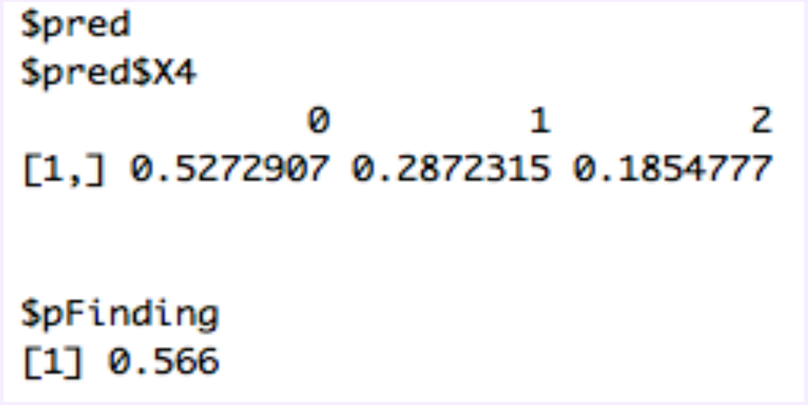
Equivalently, we could find the predicted distribution of X4 by using:

```
> xarxa.evid<-setEvidence(xarxa.estimada.gRain,
nodes=c("X1","X2"), states=c(”0",”1"))
> querygrain(xarxa.evid,nodes=c(“X4”),type=“marginal”)
```



```
$X4
X4
        0         1         2 
0.5272907 0.2872315 0.1854777 
```

We can compare the predicted conditional probability distribution of X4 to X1=0 and X2=1 with the actual values computed "by hand" from the original BN:



```
$pred
$pred$X4
             0         1         2
[1,] 0.5272907 0.2872315 0.1854777


$pFinding
[1] 0.566
```

$$
\begin{aligned}
P(X_4=0/X_1=0,\,X_2=1) &= P(X_4=0/X_3=0)\,P(X_3=0/X_1=0,\,X_2=1)\\
&\quad + P(X_4=0/X_3=1)\,P(X_3=1/X_1=0,\,X_2=1)\\
&= 0.60\times 0.80 + 0.20\times 0.20 = 0.52\,,\\
P(X_4=1/X_1=0,\,X_2=1) &= P(X_4=1/X_3=0)\,P(X_3=0/X_1=0,\,X_2=1)\\
&\quad + P(X_4=1/X_3=1)\,P(X_3=1/X_1=0,\,X_2=1)\\
&= 0.30\times 0.80 + 0.30\times 0.20 = 0.30\,,\\
P(X_4=2/X_1=0,\,X_2=1) &= 1-\left(P(X_4=0/X_1=0,\,X_2=1)+P(X_4=1/X_1=0,\,X_2=1)\right)\\
&= 1-(0.52+0.30) = 0.18
\end{aligned}
$$

They are quite similar!!

3. Constraint-Based Structure Learning

The Constraint-Based Structure Learning algorithms, also called "Conditional Independence Learners", are optimized variants of the **Inductive Causation algorithm** (Verma and Pearl, 1991).

They use statistical conditional independence tests to learn the conditional independence relationships among the variables of the model (called "constraints" in this setting) from the data.

One way to learn the structure (DAG) of a Bayesian network is to check which relationships hold according to a suitable conditional independence test.

The Inductive Causation Algorithm (IC)

Verma and Pearl (1990).

1. For each pair of variables X and Y in V, search for a set S_{XY} in V with X, Y not in S_{XY}, such that: X and Y are independent given S_{XY}, with a conditional independence test. If there is no such set, place an undirected arc between X and Y: X−Y.

   With this we learn the *skeleton* of the DAG, which is a non-directed graph.
2. For each pair of non-adjacent nodes X and Y with a common neighbour Z (that is, if we have the undirected chain X−Z−Y), check whether Z is in S_{XY}. If this is NOT true, set the direction of the arcs X−Z and Y−Z to $X \rightarrow Z$ and $Y \rightarrow Z$. The structure $X \rightarrow Z \leftarrow Y$ is called a *v-structure* (or a *collider*).
3. Set the direction of arcs which are still undirected by applying recursively the following two rules:
   i. If X is adjacent to Y and there is a strictly directed path from X to Y, then set the direction of X−Y to $X \rightarrow Y$;
   ii. If X and Y are not adjacent but $X \rightarrow Z$ and Z−Y, then change the latter to $Z \rightarrow Y$.
4. Return the resulting partially directed acyclic graph.

Conditional Independence Tests

Classic tests that are used because they are fast: the asymptotic discrete tests Kullback's Conditional Mutual Information and Pearson's χ^2, both with a χ^2 distribution. Other tests can also be used.

If X and Y are variables with a and b configurations, respectively, and **Z** is a set of variables with c configurations (globally), and we denote by n_{ijk} the observed frequency of cases with the configuration i of X, j of Y and k of **Z**, Pearson's χ^2 conditional independence test is based on the statistic:

$$\chi^2(X,Y/\mathbf{Z}) = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} \frac{\left(n_{ijk} - \frac{n_{i\cdot k}\, n_{\cdot jk}}{n_{\cdot\cdot k}}\right)^2}{\frac{n_{i\cdot k}\, n_{\cdot jk}}{n_{\cdot\cdot k}}}$$

and Kullback's Conditional Mutual Information conditional independence test is based in the statistic:

$$MI(X,Y/\mathbf{Z}) = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} \frac{n_{ijk}}{N} \log\left(\frac{n_{ijk}\, n_{\cdot\cdot k}}{n_{i\cdot k}\, n_{\cdot jk}}\right)$$

The null hypothesis (conditional independency between X and Y given **Z**) is tested using the asymptotic chi-square distribution with (a-1)(b-1)c degrees of freedom, that is, $\chi^2_{(a-1)(b-1)c}$.

More on the Inductive Causation Algorithm (IC)

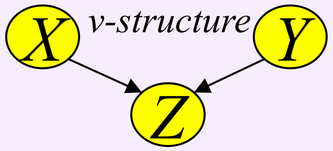
The first step in IC algorithm identifies which pairs of variables in the BN (Γ ,P) are connected by an undirected arc, regardless of its direction. These pairs of variables cannot be independent given any other subset of variables. We say that they are **d-separated**.

d-separation in the DAG Γ ⇔ Conditional independence with respect to P.

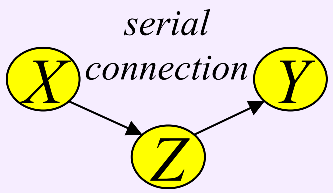
D-separation (Pearl, 1988): Let X and Y be variables in V, and **S** a subset of V. Then, **S** is said to d-separate X and Y if along every path between X and Y (a sequence of arcs disregarding arc directions) there is a node Z satisfying one of the following two conditions:

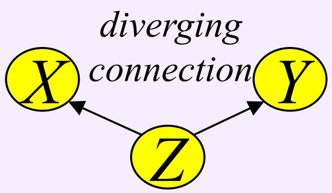
a) Z has converging arcs (that is, we have a *v-structure* or *collider* → Z ←) and none of Z or its descendants are in **S**.

b) Z is in **S** and does not have converging arcs.



a) Converging arcs: given Z, X and Y are conditionally **dependent** with respect to P, so they cannot be d-separated. Then in order to have d-separation we have to impose that Z cannot be in **S**.





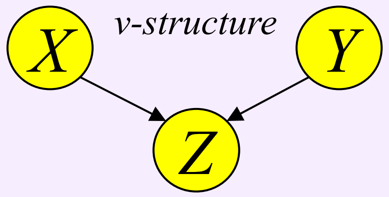
b) Not converging arcs: given Z, X and Y are conditionally **independent** with respect to P, so they have to be d-separated. Then, Z has to be in **S** for having d-separation.

More on the Inductive Causation Algorithm (IC)

The second step in IC algorithm deals with the identification of the *v-structures* among all the pairs of non-adjacent nodes X and Y with a common neighbour Z,

X–Z–Y.

By definition, *v-structures* are the only fundamental connection in which the two non-adjacent nodes are not independent conditional on the third one. Therefore, if there is a subset of nodes **S** that contains Z and that d-separates X and Y, the three nodes are part of a *v-structure* centered in Z, that is, we have:



This condition can be verified by performing a conditional independence test between X and Y with respect to every possible subset of their common neighbours that includes Z. At the end of the second step, both the *skeleton* and the *v-structures* of the network are known.

The third and last step of the IC algorithm identifies compelled arcs and orients them recursively to obtain the partially directed acyclic graph.

More on the Inductive Causation Algorithm (IC)

A major problem with IC algorithm is that the first two steps cannot be applied in the form described to any real-world problem due to the huge number of possible conditional independence relationships.

This has led to the development of improved algorithms such as the followings:

- PC (Peter & Clark): the first practical algorithm, implemented in TETRAD II, a causal discovery program developed in Carnegie Mellon University's Department of Philosophy (Scheines, Spirtes, Glymour and Meek, "TETRAD II: Tools for Discovery". Lawrence Erlbaum Associates, Hillsdale, N J, 1994). Starts with the fully connected *skeleton* model (saturated graph) in which every node is adjacent to every other node, and then a backward selection procedure is implemented.
- Grow-Shrink, GS (Margaritis, 2003).
- Incremental Association, IAMB (Tsamardinos et al., 2003).
- …

All of them except PC, first learn the Markov blanket of each node in the network.

This preliminary step greatly simplifies the identification of the neighbours of each node, as the search can be limited to its Markov blanket.

As a result, the number of conditional independence tests performed by the learning algorithm is significantly reduced!

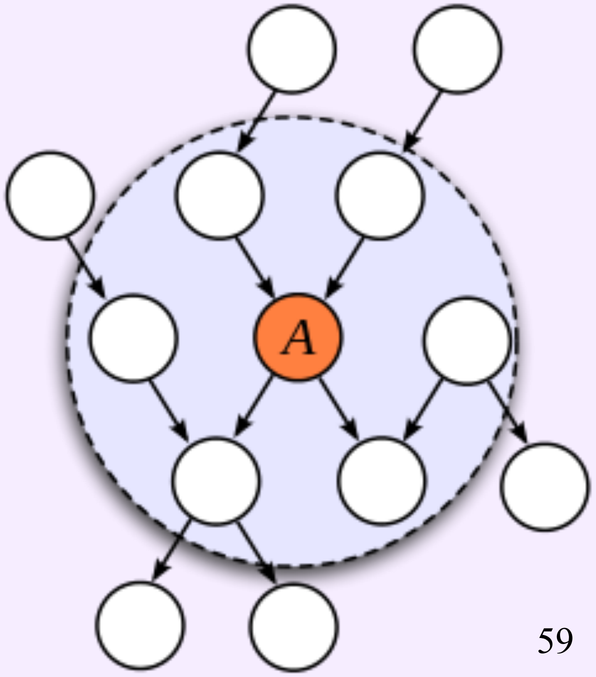
More on the Inductive Causation Algorithm (IC)

"Markov blanket" is another fundamental object closely related with that of d-separation.

Markov blanket (Pearl, 1988):
Given a node X in V, the Markov blanket of X is the set of nodes in V that d-separates X from the rest of the nodes in V.

Equivalently, the Markov blanket of a node X is the set of the parents of X, the children of X, and all other nodes sharing a child with X.

The Markov blanket of a node contains all the variables that shield the node from the rest of the network.
This means that the Markov blanket of a node is the only knowledge needed to predict the behaviour of that node.



More on the Inductive Causation Algorithm (IC)

What Verma and Pearl's IC algorithm is discovering is the set of Markov equivalent BN models (also known as statistically equivalent models).

Markov equivalence:
Two BN models are Markov equivalent if they contain the same set of variables V and any probability distribution that can be represented by one can be represented by the other.
In other words, if for all disjoints subsets of V, say **X**, **Y**, **Z**, we have that **X** is entailed to be independent of **Y** conditional to **Z** in one of the DAGs if and only if, the same happens in the other.

Then, the power of the IC algorithm follows from the following result:

Theorem (Verma and Pearl, 1990):
Any two BN models over the same set of variable V that have the same *skeleton* and the same *v-structures* are Markov equivalent.

Step 1 in the IC algorithm will recover the *skeleton*. Step 2 will recover the *v-structures*. Step 3 merely enforces consistency with DAG structure. This theorem ensures that the partially oriented graph obtained by the IC algorithm can only be completed by directing unoriented arcs in Markov equivalent ways.

More on the Inductive Causation Algorithm (IC)

Theorem (Verma and Pearl, 1990):
Any two BN models over the same set of variable V that have the same *skeleton* and the same *v-structures* are Markov equivalent.

In addition, this theorem provides a simple graphical criterion of Markov equivalence. For example,

A. All fully connected models (in the same variables) are Markov equivalent since they cannot contain *v-structures*. The reason is that the end variables in any candidate chain to be a *v-structure* must themselves be directly connected, making the *v-structure* impossible.
B. $X \rightarrow Z \rightarrow Y$ and $X \leftarrow Z \leftarrow Y$ are Markov equivalent (there is no *v-structure* here).
C. $X \rightarrow Z \rightarrow W \leftarrow Y$ and $X \leftarrow Z \rightarrow W \leftarrow Y$ are Markov equivalent (the only v-structure, centered on W, is retained).

A related important result:

Theorem (Chikering, 1995):
If (Γ_1,P_1) and (Γ_2,P_2) are two Markov equivalent BN models over the same set of variable V, then they have the same maximum likelihoods relative to any database set D, that is, $\max_{\theta^{\Gamma_1}} L^D_{\Gamma_1}(\theta^{\Gamma_1}) = \max_{\theta^{\Gamma_2}} L^D_{\Gamma_2}(\theta^{\Gamma_2})$.

4. Hybrid Structure Learning

Hybrid algorithms combine constraint-based and score-based structure learning algorithms to complement the respective strengths and weaknesses; they are considered the state of the art in current literature.
They work by alternating the following two steps:

- learn some conditional independence constraints to restrict the number of candidate networks,
- find the best network (based in a score) that satisfies those constraints and define a new set of constraints to improve on.

These two steps can be repeated several times (until convergence), but one or two times is usually enough test.

The Sparse Candidate Algorithm

Just one scheme of hybrid algorithm:

1. Choose a network structure (DAG) Γ on the variables of the model V, which usually (but not necessarily) is empty.
2. Repeat the following steps until convergence:
   a. **restrict**: for each node X_i in V, select a set C_i of candidate parents, which must include the parents of X_i in Γ.
   b. **maximise**: find the network structure Γ^* that maximizes score among the networks in which the parents of each node X_i are included in the corresponding set C_i.
3. Return the final DAG.

- ➢ Since only the general framework is defined, it is easy to modify to use newer constraint-based and/or score-based algorithms.
- ➢ It is usually faster than the alternatives, and more stable.